Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for removing dead code in code fragments of a program, comprising:

processing a first code fragment and storing first information generated during this processing indicative of whether an instruction for assigning a register in a first code fragment is possibly live, the first information including a pointer to each instruction for assigning a register that is possibly live for an exit of the first code fragment and a first register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in an instruction pointed to by a pointer included in the first information;

processing a second code fragment and storing second information generated during this processing indicative of register usage, the second information including information associated with an entry into the second code fragment;

at a time when linking the exit from the first code fragment to the entry in the ~~and~~ second code fragment ~~fragments are to be linked~~,

determining, by use of the first stored information associated with the exit and the second stored information associated with the entry, if an instruction in the first code fragment that assigns a register is a dead instruction; and

responsive to determination that an instruction is a dead instruction, eliminating the dead instruction.

2. (Original) A method according to claim 1, wherein eliminating the dead instruction comprises overwriting the dead instruction with a NOP.

3. (Original) A method according to claim 1, wherein eliminating the dead instruction comprises compacting the surrounding instructions to delete the dead instruction.

4. (Cancelled)

5. (Cancelled)

6. (Cancelled)

7. (Currently Amended) A method according to claim ~~6~~ 1, wherein the second information associated with ~~each~~ the entry includes a second register mask, the second register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in the second fragment before being read.

8. (Original) A method according to claim 7, where said determining step comprises comparing corresponding positions of the first and second register masks, wherein said eliminating step includes eliminating an instruction for assigning a register in the first code fragment if the positions corresponding to the register in the first and second register masks are both set.

9. (Original) A method according to claim 8, wherein said eliminating step further comprises determining which instruction to overwrite with reference to the pointers in first information.

10. (Original) A method according to claim 4, wherein the first information associated with each exit is stored in a epilog associated with the exit, and the second information associated with each entry is stored in a prolog associated with that entry.

11. (Currently Amended) A computer readable medium comprising instructions for removing dead code in code fragments of a program, the instructions configured to:

process a first code fragment and store first information generated during this processing indicative of whether an instruction for assigning a register in a first code fragment is possibly live, the first information including a pointer to each instruction for assigning a register that is possibly live for an exit of the first code fragment and a first register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in an instruction pointed to by a pointer included in the first information;

process a second code fragment and store second information generated during this processing indicative of register usage, the second information including information associated with an entry into the second code fragment;

at a time when linking the exit from the first code fragment to the entry in the ~~and~~ second code fragment ~~fragments are to be linked~~, determine, by use of the first stored information associated with the exit and the second stored information associated with the entry, if an instruction in the first code fragment that assigns a register is a dead instruction; and

responsive to determination that an instruction is dead instruction, eliminate the dead instruction.

12. (Original) A computer readable medium according to claim 11, wherein eliminating the dead instruction comprises overwriting the dead instruction with a NOP.

13. (Original) A computer readable medium according to claim 11, wherein eliminating the dead instruction comprises compacting the surrounding instructions to delete the dead instruction.

14. (Cancelled)

15. (Cancelled)

16. (Cancelled)

17. (Currently Amended) A computer readable medium according to claim ~~16~~ 11, wherein the second information associated with ~~each~~ the entry includes a second register mask, the second register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in the second fragment before being read.

18. (Original) A computer readable medium according to claim 17, where said determining step comprises comparing corresponding positions of the first and second register masks, wherein said eliminating step includes eliminating an instruction for assigning a register in the first code fragment if the positions corresponding to the register in the first and second register masks are both set.

19. (Original) A computer readable medium according to claim 18, wherein said eliminating step further comprises determining which instruction to overwrite with reference to the pointers in first information.

20. (Original) A computer readable medium according to claim 14, wherein the first information associated with each exit is stored in an epilog associated with that exit, and the second information associated with each entry is stored in a prolog associated with that entry.